Array cheatsheet for coding interviews

👋

Introduction

Arrays hold values of the same type at contiguous memory locations. In an array, we're usually concerned about two things - the position/index of an element and the element itself. Different programming languages implement arrays under the hood differently and can affect the time complexity of operations you make to the array. In some languages like Python, JavaScript, Ruby, PHP, the array (or list in Python) size is dynamic and you do not need to have a size defined beforehand when creating the array. As a result, people usually have an easier time using these languages for interviews.
Arrays are among the most common data structures encountered during interviews. Questions which ask about other topics would likely involve arrays/sequences as well. Mastery of array is essential for interviews!
Advantages

- Store multiple elements of the same type with one single variable name
- Accessing elements is fast as long as you have the index, as opposed to linked listswhere you have to traverse from the head.

Disadvantages

- Addition and removal of elements into/from the middle of an array is slow because the remaining elements need to be shifted to accommodate the new/missing element. An exception to this is if the position to be inserted/removed is at the end of the array.
- For certain languages where the array size is fixed, it cannot alter its size after initialization. If an insertion causes the total number of elements to exceed the size, a new array has to be allocated and the existing elements have to be copied over. The act of creating a new array and transferring elements over takes O(n) time.

Learning resources

- Readings
  - Array in Data Structure: What is, Arrays Operations, Guru99
- Videos
  - Arrays, University of California San Diego

Common terms

Common terms you see when doing problems involving arrays:

- Subarray - A range of contiguous values within an array.
  - Example: given an array `[2, 3, 6, 1, 5, 4]`, `[3, 6, 1]` is a subarray while `[3, 1, 5]` is not a subarray.
- Subsequence - A sequence that can be derived from the given sequence by deleting some or no elements without changing the order of the remaining elements.
  - Example: given an array `[2, 3, 6, 1, 5, 4]`, `[3, 1, 5]` is a subsequence but `[3, 5, 1]` is not a subsequence.

Time complexity

Operation	Big-O	Note
Access	O(1)	
Search	O(n)	
Search (sorted array)	O(log(n))	
Insert	O(n)	Insertion would require shifting all the subsequent elements to the right by one and that takes O(n)
Insert (at the end)	O(1)	Special case of insertion where no other element needs to be shifted
Remove	O(n)	Removal would require shifting all the subsequent elements to the left by one and that takes O(n)

Remove (at the end)	O(1)	Special case of removal where no other element needs to be shifted

Things to look out for during interviews

- Clarify if there are duplicate values in the array. Would the presence of duplicate values affect the answer? Does it make the question simpler or harder?
- When using an index to iterate through array elements, be careful not to go out of bounds.
- Be mindful about slicing or concatenating arrays in your code. Typically, slicing and concatenating arrays would take O(n) time. Use start and end indices to demarcate a subarray/range where possible.

Corner cases

- Empty sequence
- Sequence with 1 or 2 elements
- Sequence with repeated elements
- Duplicated values in the sequence

Techniques

Note that because both arrays and strings are sequences (a string is an array of characters), most of the techniques here will apply to string problems.

Sliding window

Master the sliding window technique that applies to many subarray/substring problems. In a sliding window, the two pointers usually move in the same direction will never overtake each other. This ensures that each value is only visited at most twice and the time complexity is still O(n). Examples: Longest Substring Without Repeating Characters, Minimum Size Subarray Sum, Minimum Window Substring

Two pointers

Two pointers is a more general version of sliding window where the pointers can cross each other and can be on different arrays. Examples: Sort Colors, Palindromic Substrings
When you are given two arrays to process, it is common to have one index per array (pointer) to traverse/compare the both of them, incrementing one of the pointers when relevant. For example, we use this approach to merge two sorted arrays. Examples: Merge Sorted Array

Traversing from the right

Sometimes you can traverse the array starting from the right instead of the conventional approach of from the left. Examples: Daily Temperatures, Number of Visible People in a Queue

Sorting the array

Is the array sorted or partially sorted? If it is, some form of binary search should be possible. This also usually means that the interviewer is looking for a solution that is faster than O(n).
Can you sort the array? Sometimes sorting the array first may significantly simplify the problem. Obviously this would not work if the order of array elements need to be preserved. Examples: Merge Intervals, Non-overlapping Intervals

Precomputation

For questions where summation or multiplication of a subarray is involved, pre-computation using hashing or a prefix/suffix sum/product might be useful. Examples: Product of Array Except Self, Minimum Size Subarray Sum, LeetCode questions tagged "prefix-sum"

Index as a hash key

If you are given a sequence and the interviewer asks for O(1) space, it might be possible to use the array itself as a hash table. For example, if the array only has values from 1 to N, where N is the length of the array, negate the value at that index (minus one) to indicate presence of that number. Examples: First Missing Positive, Daily Temperatures

Traversing the array more than once

This might be obvious, but traversing the array twice/thrice (as long as fewer than n times) is still O(n). Sometimes traversing the array more than once can help you solve the problem while keeping the time complexity to O(n).

Essential questions

These are essential questions to practice if you're studying for this topic.

- Two Sum
- Best Time to Buy and Sell Stock
- Product of Array Except Self
- Maximum Subarray

Recommended practice questions

These are recommended questions to practice after you have studied for the topic and have practiced the essential questions.

- Contains Duplicate
- Maximum Product Subarray
- Search in Rotated Sorted Array
- 3Sum
- Container With Most Water
- Sliding Window Maximum